Description of assignments

By Sadaf, Rumyana, and Christian

Assignment 1

We modified the complexity visitor by changing the idea of complexity from number of if-statements to number of import statements. We differ between ‘import’ and ‘import from’ statements.

This definition of code complexity will give a nice overview of the most and least used libraries, so if a lot of your files depend on some outdated third-party library you may want to change this to ensure future quality of the code.

This is implemented by creating a dictionary for each type of import statements where the library is a key and the value is the count of files this is imported in (it will count identical import statements in the same file twice).

The compute_complexity function is modified to return a tuple of values (the two dictionaries). We add visit functions for the two import statements that are similar to the visit_if function. If the dictionary item does not exist, add it with a value of 1, otherwise increment the value of the pre-existing item.

We make use of glob to efficiently read multiple files in a folder (can also be used for multiple folders and with opportunity to only include .py-files.

Assignment 2

For the second assignment we wish to compute the fan-out for every function in a given file. We do this by adding a new dictionary with a key and the type of value is a list (key=function, value=all outgoing method calls in function).

We create a new compute function called compute_fanout which traverses the AST and returns the items in the dictionary.

We create a visit_Call function to identify and add the outgoing method calls of a given function to the dictionary. However, we have not figured out how to get the name of the outgoing method call which is why the function currently just adds it’s own name for every outgoing method call.